Credit Card Default Prediction

Architecture Design

Devarshi Choudhury

Contents

1. **Introduction**.....2

1.1**. What is an Architecture Design Document?** 2

1.2. **Scope**.....2

1.3 **Proposed Solution** 3

2. **Workflow Architecture**.....4

3. **Architecture Description**.....5

3.1. **Data Description**.....5

3.2. **Data Transformation**.....5

3.3. **EDA (Exploratory Data Analysis)** 5

3.4. **Data Insertion into Database**.....5

3.5. **Export Data from Database**.....5

3.6. **Data Cleaning** 5

3.7. **Data Pre-processing** 6

3.8. **Feature Engineering**.....6

3.9. **Feature Selection**.....6

3.10. **Model Building** 6

3.11. **Random Forest Model** 6

3.12. **LightGBM Model**.....6

3.13. **XGBoost Model** 7

3.14 **Web Application with Streamlit** 7

3.14.1 **User Interface Elements** 8

3.14.2 **Styling and Layout** 8

3.14.3 **Interactive Feature** 8

3.15. **Deployment**.....8

3.16 **Technologies Used** 8

4. **Conclusion** 8

1. Introduction

1.1 What is an Architecture Design Document (ADD)?

This Architecture Design Document (ADD) provides an overview of the architecture of the Credit Card Default Prediction Web Application. The application is designed to predict the likelihood of a credit card user defaulting on their payments for the next month based on various input parameters such as demographics, payment history, and bill amounts.

1.2 Scope

The scope of work for the Credit Card Default Prediction Web Application includes the following:

a. Frontend Development

- **User Interface Design**: Designing a user-friendly interface using Streamlit.
- **Input Form**: Creating input forms for users to enter their demographic information, repayment status, bill amounts, and previous payment amounts.
- **Prediction Display**: Displaying the prediction result, probability of default, and credit utilization ratio.

b. Backend Development

- **Model Training**: Training a Random Forest Classifier model using historical credit card data.
- **Preprocessing Module**: Developing a module to preprocess user input data, including encoding categorical variables and calculating the credit utilization ratio.
- **Prediction Module**: Implementing a module to utilize the trained model for making predictions.
- **Model Loading**: Developing a module to load the trained model from disk.

c. Data Flow

- **Data Preprocessing**: Handling missing values, encoding categorical variables, and calculating the credit utilization ratio.
- **Model Prediction**: Utilizing the trained model to predict the likelihood of default based on user input.
- **Displaying Results**: Displaying prediction results, probability of default, and credit utilization ratio on the web interface.

d. Testing and Validation

- **Unit Testing**: Testing individual components such as model loading, preprocessing, and prediction modules.
- **Integration Testing**: Ensuring proper integration of frontend and backend components.
- **Validation**: Validating the prediction results against known outcomes to ensure accuracy.

e. Deployment

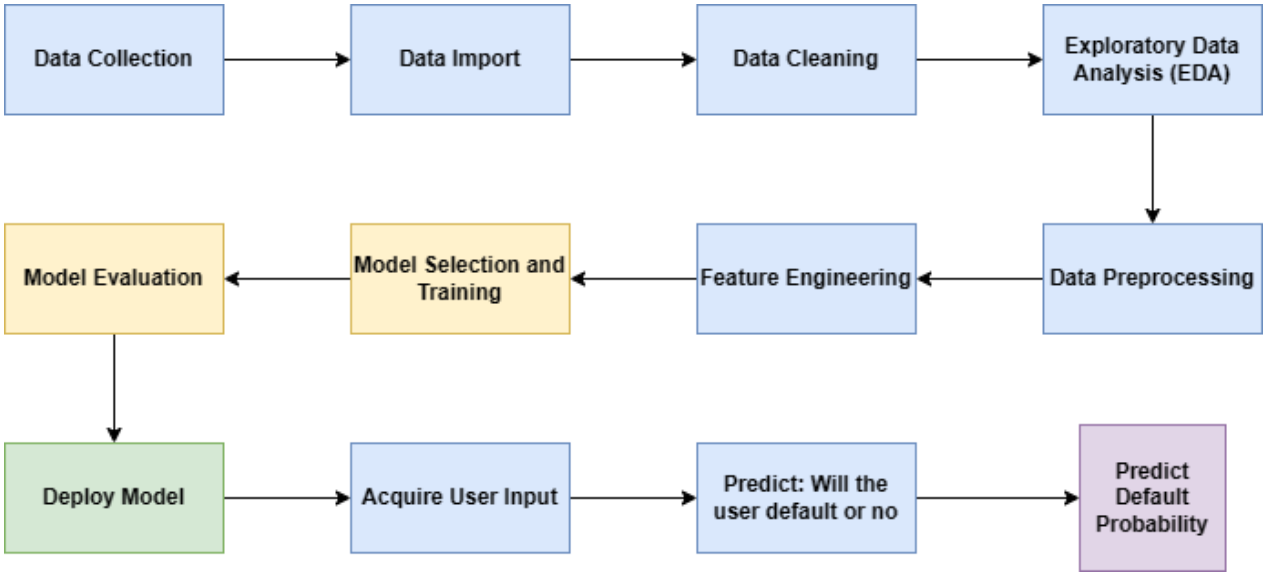
- **Platform Selection**: Selecting a suitable platform for deployment such as Heroku, AWS, or Streamlit.
- **Configuration**: Configuring the environment with required dependencies and libraries.
- **Deployment**: Deploying the application to the selected platform for public access.

1.3 Proposed Solution

The proposed solution is a web application that serves as an intuitive interface for users to input customer details. These details are then processed by a trained machine learning model running in the backend. The model predicts the likelihood of a customer defaulting on their credit card payments. The web application displays both the prediction outcome and the corresponding probability of default on the user-facing frontend page, providing users with valuable insights for decision-making.

2. Workflow Architecture

a. **Frontend:** Streamlit for creating the user interface and app deployment.
b. **Backend:** Python for data processing, model training, and prediction.



3. Architecture Description

3.1 Data Description:

The dataset used in this project is the "Default of Credit Card Clients Dataset" obtained from Kaggle:

- Data Source: Default of Credit Card Clients Dataset
- This dataset contains 25 attributes with 30,000 observations for both training and testing sets.
- Key features include credit limit, gender, education level, marital status, age, payment status for multiple months, and bill amounts.

3.2 Data Transformation:

- The original CSV dataset is converted into a pandas DataFrame.
- After reading the dataset, two DataFrames train and test are concatenated into one DataFrame for further processing.

3.3 Exploratory Data Analysis (EDA):

- EDA is conducted to investigate the dataset, identify patterns, anomalies (outliers), and form hypotheses.
- It involves statistical analysis, visualizations, and summary statistics to gain insights into the data.

3.4 Data Insertion into Database:

- Creation of a database and connection establishment.
- Table creation within the database to store the dataset.
- Insertion of dataset into the database for storage and retrieval.

3.5 Export Data from Database:

- The stored dataset is exported from the database as a CSV file.
- This exported CSV file is used for further data preprocessing and model training.

3.6 Data Cleaning:

- Data cleansing involves detecting and correcting (or removing) corrupt or inaccurate records.
- Tasks include handling missing values, correcting data types, and removing duplicates.

3.7 Data Pre-processing:

- Pre-processing prepares the raw data for machine learning models.
- Steps include scaling numerical features, encoding categorical variables, and handling imbalanced data.

3.8 Feature Engineering:

- Feature engineering is utilized to create new features that enhance model performance.
- In this project, features like 'Credit Utilization Ratio', 'Age Groups', and 'Payment Status Trends' are engineered from existing attributes.

3.9 Feature Selection:

- Feature selection is crucial to improve model efficiency and reduce overfitting.
- Techniques like Variance Inflation Factor (VIF) are employed to detect and potentially remove highly correlated features.

3.10 Model Building:

- Machine learning models are built using various algorithms such as Random Forest, LightGBM, and XGBoost.
- Training the models involves splitting the data into training and testing sets, fitting the model, and evaluating its performance.

3.11 Random Forest Model:

- Random Forest is a meta-estimator that fits multiple decision tree classifiers on random sub-samples of the dataset.
- It improves predictive accuracy by averaging results and helps control overfitting.

3.12 LightGBM Model:

- LightGBM is a gradient boosting framework that uses tree-based learning algorithms.
- It is known for its speed and efficiency, particularly with large datasets.
- LightGBM builds trees vertically, leaf-wise rather than level-wise, reducing loss more efficiently.

3.13 XGBoost Model:

- XGBoost is an optimized distributed gradient boosting library.
- It is highly efficient, scalable, and flexible, making it a popular choice for machine learning competitions.
- XGBoost uses a gradient boosting framework and is known for its speed and performance.

These models (Random Forest, LightGBM, XGBoost) were trained and evaluated to predict credit card defaults in the project. Each model has its strengths and was tested to determine the best performing one for the given dataset.

3.14 Web Application with Streamlit:

- Streamlit is used to create the web application interface.
- The app includes input widgets for user input, prediction functionality, and displays the prediction results.
- Users can input features like credit limit, gender, education, marital status, age, payment status, and bill amounts.
- Upon entering the input values, the application predicts in real-time whether the customer is likely to default

3.14.1 User Interface Elements:

- Sidebar containing input parameters for demographic information.
- Input fields for credit limit and age.
- Select boxes for gender, education level, and marital status.
- Dropdown menus for repayment status and bill amounts.
- Cross button on the sidebar allows users to hide the input parameter sidebar.

3.14.2 Styling and Layout:

- Custom colour theme of light green and white for a visually appealing interface.
- Option to switch between dark mode and light mode for user preference.
- Responsive design for cross-platform compatibility and various screen sizes.

3.14.3 Interactive Features:

- Real-time updates based on user input.
- Hover effects on elements for visual feedback.
- Dynamic display area adjusts based on user actions for a seamless interaction flow.

3.15 Deployment:

- The web application can be deployed using platforms like Heroku, AWS, or Streamlit Sharing.
- Deployment involves setting up a web server to host the app and making it accessible via a URL.

3.16 Technologies Used:

- Python: Main programming language for backend development.
- Streamlit: Web application framework for creating the UI.
- Pandas: Data manipulation and preprocessing.
- Scikit-learn: Machine learning library for model training and evaluation.

4. Conclusion

This architecture description provides a comprehensive overview of the data processing, model building, and frontend components used in the Credit Card Default Prediction project. It outlines the steps from data transformation to model deployment, highlighting the technologies and methodologies employed throughout the project.